Performance optimization on Shared Hosting

1).Enabling cache module:-

When you look to improve performance of your Drupal 7 site when it is on a shared web hosting plan, you are limited in options due to the nature of shared hosting. Many of the performance optimization options you would have if you were on a Virtual Private Server (VPS) or on your own dedicated hosting box are not available.

Some of the reasons why you are limited in a shared web hosting plan:

- No configuring the web server software
- No configuring the relational DB
- No extra services like APC (Alternative PHP Cache) or Memcached that allow you fetch objects from fast memory instead of slow hard disks
- No consistency with service, you are sharing with lots of people so you typically don't have dedicated hardware resources for running your site
- No easy path to upgrade hardware resources within shared hosting, typically need to move to a different service like a VPS or dedicated hosting as next step

Yes, these are pretty limiting, and if you can afford it, I definitely recommend going with at least a VPS if your web presence is a large part of your business. However, if you just have a blog or a brochure-type site, shared hosting can be just fine. With all these limitations, you still have a lot you can do to performance tune your Drupal site on shared web hosting plans. The options below are listed in order of the amount of performance impact for the implementation time they typically have on a shared hosting Drupal site.

1) Turn Page Caching On

What page caching does is that instead of using a bunch of database queries to get the data used in making a typical web page, the rendered contents of the web page are stored in a separate database cache table so that it can be recalled quicker. If you have 10 people visiting the site from different computers, Drupal first looks into the database cache table to see if the page is there, if it is, it just gives them the page. Think of saving the output of 50 separate queries so that is

accessible with a single query. You obviously are reducing the SQL queries required by a lot. What the page cache table actually stores is HTML content.

One gotcha with this Page Caching is that it only works to optimize the page load time for Anonymous users. This is because when you are logged in, you might have blocks that show up on the page that are customized for you, if it served everybody the same page, they would see your customized information (think of a My Recent Posts block), so Drupal does not use the Page Cache for Authenticated users automatically. This allows you to turn Page Caching on and still get the benefit for Anonymous user page load times, but does not break the site for Authenticated users. There are other caching options that will help with Authenticated user page performance, we will talk about those later.

To enable Page Caching, you go to Configuration > Development > Performance and select the checkbox next to "Cache pages for anonymous users". See screenshot:

An alternative to the page caching built into Drupal 7 is to use Boost to cache your pages as static HTML files instead of in the database, which can be served very efficiently especially in shared hosting environments. If you use Boost, you would not want the Drupal page caching enabled like mentioned above. Boost would be a preferred option except for the fact that currently, Boost is still in beta for Drupal 7 even though Drupal 7 has been out for quite a while, and I don't often recommend using beta modules unless you have tested them thoroughly. Also, like the Drupal page caching, this still only works for Anonymous users.

2) Aggregate and Compress CSS Files

Drupal is very flexible, but with all that flexibility, you will have some performance tradeoffs. One of the ways these tradeoffs are easy to see is in the number of CSS files you see loaded in each Drupal page. There can be 30 or more CSS files even for pretty basic Drupal sites, the reason for this is that Drupal operates using modules, and whenever a module has some CSS files that it wants to add, it is a separate file. Your theme is also going to provide several CSS files.

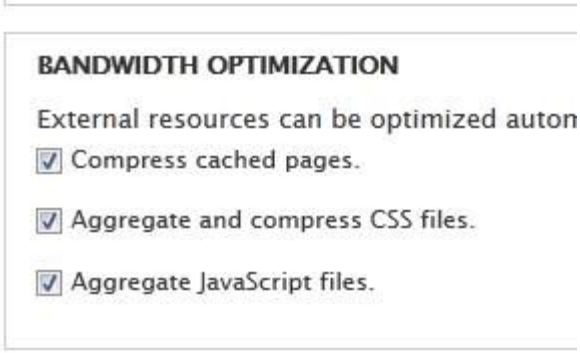
This can cause performance problems because you have to make 30 separate requests to the web server, one for each file. We would like to reduce the number of requests to the web server as a general rule of web performance optimization.

So by enabling "Aggregate and compress CSS files", Drupal will take your CSS files and reduce the number of them and size of them. The Aggregate part of that is that instead of having 30 separate files, Drupal will concatenate all the files together and separate them into just a few files. It doesn't reduce them to 1, but reduces it down to 8 files or so. The Compression part of that setting will remove all whitespace in the CSS files, those whitespaces each contribute to the size but have no functional purpose. So this setting reduces the number and total size, which will increase the speed of loading CSS.

3) Aggregate JS

Just like Aggregate and Compress CSS files, you can aggregate Javascript files with this setting. For the same reason why there are many CSS files, there can be many JS files, each module can provide one or many of its own Javascript files. Aggregation reduces the number of files down to around 8. So by Aggregating both CSS and Javascript files, you can reduce the requests from 60+ to less than 20, this can help to deal with the shared resources so you can get as much from the server with each request as possible.

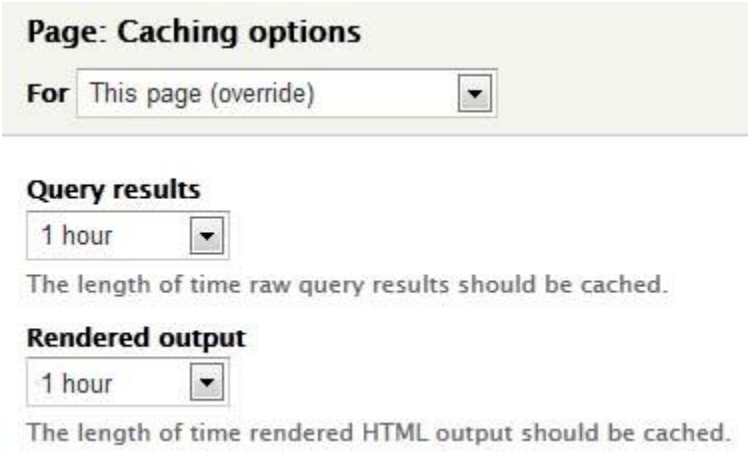
For items 2), and 3), the Settings can be enabled in Drupal 7 on the Configuration | Development | Performance age, you see the Bandwidth Optimization section by checking all three checkboxes:



4) Turn Views caching on

As mentioned when talking about Page Caching only working for anonymous users above, there are other caching options for helping with Authenticated user page performance. One of those options is to turn on caching for blocks and pages that you create using the Views module. This allows you to cache the output of the query used to generate the view, or the end HTML output of your View, and you can tune the cache for them separately. And realize too that this means you can cache portions of a page if you are using one or several Views blocks in the page, it will just cache that block in the page, not the whole page.

You can find the settings for caching a View when you are in the Views UI and edit your view, then under Advanced | Other | Caching after you click Time-based to enable caching, you can see the settings below where you can specify the maximum age of a cached query or page:



The way that specifying the maximum age works is that when the cached copy is stored in the database, every time that a user visits the page with the View in it, it

will check to see if the cached copy is too old and will fetch a fresh copy if it is expired. This is the nature of caching, you have to determine for the typical use of your site what kind of cache usage you can have. If changes are made to your site infrequently, then the cache doesn't have to be cleared often. If changes are made often, it may be best to keep the maximum age times pretty low so that people see a View that represents the actual contents of the site that will be out of date by no more than 1 hour (using the settings above as an example).

The difference between caching the Query results and the Rendered output applies more when you are doing some Theme altering, you may not want to cache the rendered output, but could instead just cache the query results. The Rendered output is achieved using the Query results, so you can likely save a bulk of the time using Query results alone. Bottom line, if you can use Rendered output caching, it will give you the best caching results.

5) Turn block caching on

This is another type of caching to help out with Authenticated user performance. This will cache individual blocks, not pages, but since many blocks can be used to build up a page, doing block caching can still be a significant performance improvement. This block caching is not the same as caching a Views block, in fact, it does not apply to Views blocks. This block caching only applies to custom blocks that you create, or blocks that modules provide. Modules themselves provide what type of caching is allowed for blocks to help do it appropriately, as some blocks should be cached per user, some cached per role, all meaning that for a single role, a block might be the same for all users in that role, etc.

To enable Block Caching, you go to Configuration | Development and select the checkbox next to "Cache blocks". See screenshot:

6) Disable Modules You Are Not Using

Having unused modules that are enabled adds to overhead for additional PHP code to execute on each page load, extra CSS and Javascript files included with each page load, even if you are not using the module for anything.

Going one step further, if you have a module that you are only using one small part of, and can achieve the same results using another method utilizing an existing module or another method in core Drupal, then that's always a good idea for optimizing performance.

7) Move Images, Videos, and Static Files to a CDN (Content Delivery Network)

Offloading static files to another server is a way to conserve resources on your shared hosting. This helps with avoiding disk I/O and reduces the number of requests that your shared hosting has to handle. One common strategy is to use a CDN (content delivery network) to host some of your static files, with images and videos being one of the easiest ways to implement this, especially if they are a part of your theme.

Since web browsers restrict the number of requests a single domain name can handle in parallel at any given moment to 4-6, if you move some of your assets to another domain, you can request another 4-6 in parallel from that domain, so it can increase speed in yet another way. Always a good idea if you can afford it, as CDN's usually do cost a little bit of money, the most common as of today is

Amazon S3. The cost can be really small though, like less than $5 to $10/month, check out the Amazon S3 price calculators to make sure you understand what costs you might have.

To enable this a little easier, there is a AmazonS3 module for Drupal to move the file system to S3 on Drupal 7, and other modules like the Video module integrate with the AmazonS3 module to allow videos to be auto-stored for you.

8) Implement Caching in Your Own Modules

If you are running a Drupal site where you have created your own modules, make sure to implement caching so you can take advantage of the caching system Drupal has in place. If your module will be caching a lot of different types of data, consider creating your own caching table, then you can control the maintenance of that cache table easier. Also helps if at some point in the future you move onto a non-shared hosting system where you can take your own caching tables and move them to use memcached, but leave the other cache types to use the database cache table.

9) Make Sure Your Own Modules Use Indexes Properly

If you have database queries in modules that you have developed yourself, verify that you have set up indexes properly to help those queries execute efficiently. You can use the EXPLAIN command in MySQL to see if your query is using an index, and how it is using that index. It's very helpful to use the Drupal Devel module to figure out which database queries are taking a long time. Indexes are one thing that you can control in a MySQL database even when you are on a shared hosting platform

10) Reduce Image Sizes and Number of Images

Reducing image sizes and/or number of images works to improve performance in a similar way as the CSS and JS aggregation did, you are reducing the number of requests to the server, and reducing the overall size that needs to be transferred from the server to your user's computer.

You can always work on reducing the size of images that you have by using different levels of compression. You can use software tools to try and compress

pngs, or just use settings when you save an image within Photoshop for JPEG images.

Reducing the number of images can be accomplished by using CSS image sprites to combine multiple images into a single image. This works especially well if you are using images as backgrounds for hover/active states for tabs or buttons.

2). Installing alternate modules for cache and configuration Mobile performance on responsive:-

Another great way to improve site performance is the Boost module (Note: this will only be advantageous for users who don't log into the site). Boost will cache pages as compressed html files when a user visits a page and then serve the cached version to subsequent visits to the page. The downside to the module is that it will not work straight out of the box; it will require a small amount of additional setup in the .htaccess file. The Module works well with the standard Bandwidth Optimization mentioned in the previous section, however will not play nice with the Caching, Make sure this is disabled!
The module page for Boost has a useful guide for getting everything set up.